Claims

What is Claimed is:

1. A system, comprising:

a receiving module to receive a request to load a component;

a stack to record the request;

a loader to fulfill the request, wherein when the request has been fulfilled the request is removed from the stack and when the loading of the component is unsuccessful, contents of the stack are made available to a user to indicate the unsuccessfully loaded component.

2. The system according to claim 1, further comprising:

an execution module to execute instructions contained in a component object, wherein the component object is created from information in the component.

3. The system according to claim 1, wherein the receiving module, the stack and the loader reside on a development platform.

4. The system according to claim 1, wherein the receiving module, the stack and the loader reside on an embedded device.

5. The system according to claim 4, wherein the component resides external to the embedded device.

6. The system according to claim 1, wherein the loader includes a plurality of loading modules in a hierarchical relationship and the component is loaded by a highest level loading module capable of loading the component.

7. The system according to claim 1, wherein the stack contents are made available to the user via one of an on-screen display, a printout and a file.

8. A method of loading, comprising the steps of:

receiving a request to load a first software module;

placing a representation of the first software module onto a stack;

determining if the first software module is dependent on a second software module;

placing, when the first software module is dependent on the second software module, a representation of the second software module onto the stack;

loading the second software module;

removing the representation of the second software module from the stack when the second software module has been successfully loaded;

loading the first software module; and

removing the representation of the first software module from the stack when the first software module has been successfully loaded.

9. The method according to claim 8, further comprising the step of:

making contents of the stack available to a user when the loading of one of the first software module and the second software module has been unsuccessful.

10. The method according to claim 8, wherein the stack contents are made available to the user via one of an on-screen display, a printout and a file.

11. The method according to claim 8, wherein the representation of the first software module is the name of the first software module.

12. A system comprising:

a stack to record a load request for a software component; and

a loader to receive and fulfill the load request for the software component, wherein the loader pushes a representation of the software component onto the stack when the load request is received and pops the representation of the software component off of the stack when the load request has been successfully fulfilled.

13. The system according to claim 12, wherein contents of the stack are made available to a user when fulfillment of the load request has been unsuccessful.

14. The system according to claim 12, further comprising:

an execution module to execute instructions contained in a component object, wherein the component object is created from information in the software component.

15. The system according to claim 12, wherein, when the software component in the load request is dependent on an additional software component, an additional load request for the additional software component is received by the loader and the loader pushes a representation of the additional software component onto the stack when the additional load request is received and pops the representation of the additional software component off of the stack when the

additional load request has been successfully fulfilled.

16. The system according to claim 15, wherein contents of the stack are made available to a user when fulfillment of the additional load request has been unsuccessful and the contents of the stack include the representation of the software component and the additional software component.

17. A method of loading a Java class, comprising the steps of:

receiving a request to load a first Java class;

placing a representation of the first Java class onto a stack;

determining if the first Java class is dependent on a second Java class;

placing, when the first Java class is dependent on the second Java class, a representation of the second Java class onto the stack;

loading the second Java class;

removing the representation of the second Java class from the stack when the second Java class has been successfully loaded;

loading the first Java class; and

removing the representation of the first Java class from the stack when the first Java class has been successfully loaded.

18. The method according to claim 17, further comprising the step of:

making contents of the stack available to a user when the loading of one of the first Java class and the second Java class has been unsuccessful.

19. The method according to claim 18, wherein the stack contents are made available to the user via one of an on-screen display, a printout and a file.

20. The method according to claim 17, wherein the loading steps, the placing steps and the removing steps are performed by a Java class loader.

21. The method according to claim 17, further comprising the steps of:

creating a first Java class object from the loaded first Java class; and

executing instructions included in the first Java class object.

22. The method according to claim 21, wherein the executing step is performed by a Java Virtual Machine.

23. A system, comprising:

a stack to record a load request for a Java class;

a Java class loader to receive and fulfill the load request for the Java class, wherein the loader pushes a representation of the Java class onto the stack when the load request is received and pops the representation of the Java class off of the stack when the load request has been successfully fulfilled; and

a Java Virtual Machine to execute instructions contained in a Java class object, wherein the Java class object is created from information in the Java class.

24. The system according to claim 23, wherein the Java class loader is one of a custom class loader and a default loader.

25. The system according to claim 23, wherein contents of the stack are made available to a user when fulfillment of the load request has been unsuccessful.

26. The system according to claim 23, wherein the stack, the Java class loader and the Java Virtual Machine reside on a development platform.

27. The system according to claim 1, wherein the stack, the Java class loader and the Java Virtual Machine reside on an embedded device.

28. The system according to claim 23, wherein the load request is received via a uniform resource locator.